UNIVERSITY OF SASKATCHEWAN

Department of Computer Science

CMPT 250.6 MIDTERM EXAMINATION 1

November 26th, 2001

Total Marks: 75 **CLOSED BOOK and CLOSED NOTES**

Time: 80 minutes

Instructions

Read each question carefully and write your answer legibly in an exam booklet. The marks for each question are as indicated. Allocate your time accordingly.

1. **True or False** ***(1 mark each = 11 marks)***

   In your exam booklet, clearly indicate which part is being answered and whether your answer is *true* or *false.*

   (1) The validity of software refers to the extent to which it satisfies the requirements given in the specification document. T

   (2) The type of a reference variable must always be an effective (non-deferred) class. F

   (3) When declaring a variable of a reference type in Eiffel, it consumes the same amount of space regardless of the type of this variable T

   (4) Because parameter passing in Eiffel is by value, a procedure cannot change in any way any actual argument passed into it T

   (5) Although performing timing analysis for average cases is often much harder than performing timing analysis for worst cases, the former is usually more useful when describing the timing complexity of an algorithm T

   (6) If the timing complexity of an algorithm belongs to **Θ(f(n))**, it's timing complexity also belongs to **O(f(n)*n)** T

   (7) In Eiffel, preconditions are specified by *ensure* or *ensure then*, and postconditions are specified by *require* or *require else* F

   (8) In the Design by Contract view of programming, the supplier has obligations, but the client does not F

Suppose we have a deferred class `Shape`. `Line` is an effective subclass of Shape, where Shape has a procedure `draw` that is redefined within Line. Also, suppose that `Line` adds a new procedure `dashed`:

```
local
s : Shape
do
     create{Line} s.make(x1, y1, x2, y2)
     s.draw
     s.dashed
end
```

(9) "`s.draw`" will call the procedure `draw` defined in `Line`

(10) "`s.dashed`" will call the procedure `dashed` defined in `Line`

(11) If properly used, Object-Oriented Design can generate a more maintainable, more reusable, and more reliable design than Structured Design

2. (***15 marks***) Assume for some application, the capability is needed to reverse a linked list. A good approach to this problem is to define a class with a procedure to do the job. Give the Eiffel code to define a descendant class of `LINKED_LIST_UOS`, where the descendant has a procedure called `reverse` that will reverse the current list. Since `LINKED_LIST_UOS` is an iterated class, you should consider using the iterator to implement the routine `reverse`. Note that the iterator in `LINKED_LIST_UOS` can only go forwards. Moreover, a routine named `delete_item` automatically moves to the next item when deleting an item. To aid you, the inheritance diagrams for `LINKED_LIST_UOS` and `LINEAR_ITERATOR_UOS` are given at the end of the examination paper. (These were copied from pages 222 and 224 of the text.)

3. (**10 marks**) Using the axiomatic approach, the semantics for the `SIMPLE_LIST` ADT has `make` and `insert_first` as build operations, and the following axioms:

```
p.make.is_empty = true
p.insert_first(i).is_empty = false
p.insert_first(i).first_item = i
p.insert_first(i).delete_first = p
```

To this specification add two more operations: `replace_first` and `max`. The function `replace_first(value)` returns a list identical to the target list except that the first item is replaced by `value`. The `max` operation returns the largest value in the list, assuming that the list contains **positive** integer values. Because of this assumption, the value 0 is to be returned by max for an empty list. For each of these two operations, give its precondition and the axioms needed to define it. You may assume the existence of a `max_int` function with two integer arguments that returns the larger argument.

4. ***(12 marks)*** The following routine `SearchSubString` searches for the first occurrence of string `sub` in the string `str`.
   (a) Do the timing analysis for the **worst case.**
   (b) Do the timing analysis for the **best case.**
   (c) For <u>one</u> of the cases, give an example (specific strings) that achieves the bound of the analysis.

```
SearchSubString(sub:STRING; str:STRING) : Integer is
local
  i, j : INTEGER
  found : BOOLEAN
do
  from
    i:=1
    found := false
  until i>str.count-sub.count+1 or found
  loop
    from j:=1
    until j>sub.count or sub.item(j)/=str.item(i+j-1)
    loop
      j := j+1
    end
    if j>sub.count then
      found := true
    else
      i := i+1
    end
  end
  if found then
    Result := i
  else
    Result := -1
  end
end
```

5. Consider the ROBOT game done for assignment #3. Suppose that the player of the game is given another command: list weapons in use. The purpose of this command is to display all the weapons currently held by any of the robots, including the player robot. The output is to be just the weapons, in any order. In particular, it is NOT divided up into parts according to the robots. In fact, it should not contain the same weapon twice, even though some weapon may be held by several robots.

   (a) (***15 marks***) Given a *sequence diagram* for this command, starting with the prompt from the game for the user to enter the next command. Your sequence diagram should use the classes that you used for the assignment or the classes that we used for our solution, plus any new ones that you need. In particular, the objective is to add the new functionality to the assignment solution, not to redesign it. For your reference, the following is a list of the classes used in our solution, with indentation used to show inheritance.

```
ROOT_CLASS
   PLAY_GAME
FIELD
   BATTLE_FIELD
WEAPON
ROBOT
   COMBAT_ROBOT
         PLAYER
         ROOKIE_ROBOT
         VERTERAN_ROBOT
```

   (b) (***7 marks***) What new features (functions, procedures, and attributes) are needed in the classes that you used for the assignment?

   (c) (***5 marks***) What new classes/containers did you add? What operations are implied by your sequence diagram for each new class or container? If you added any new containers, what data structures would you suggest for the containers?

The End

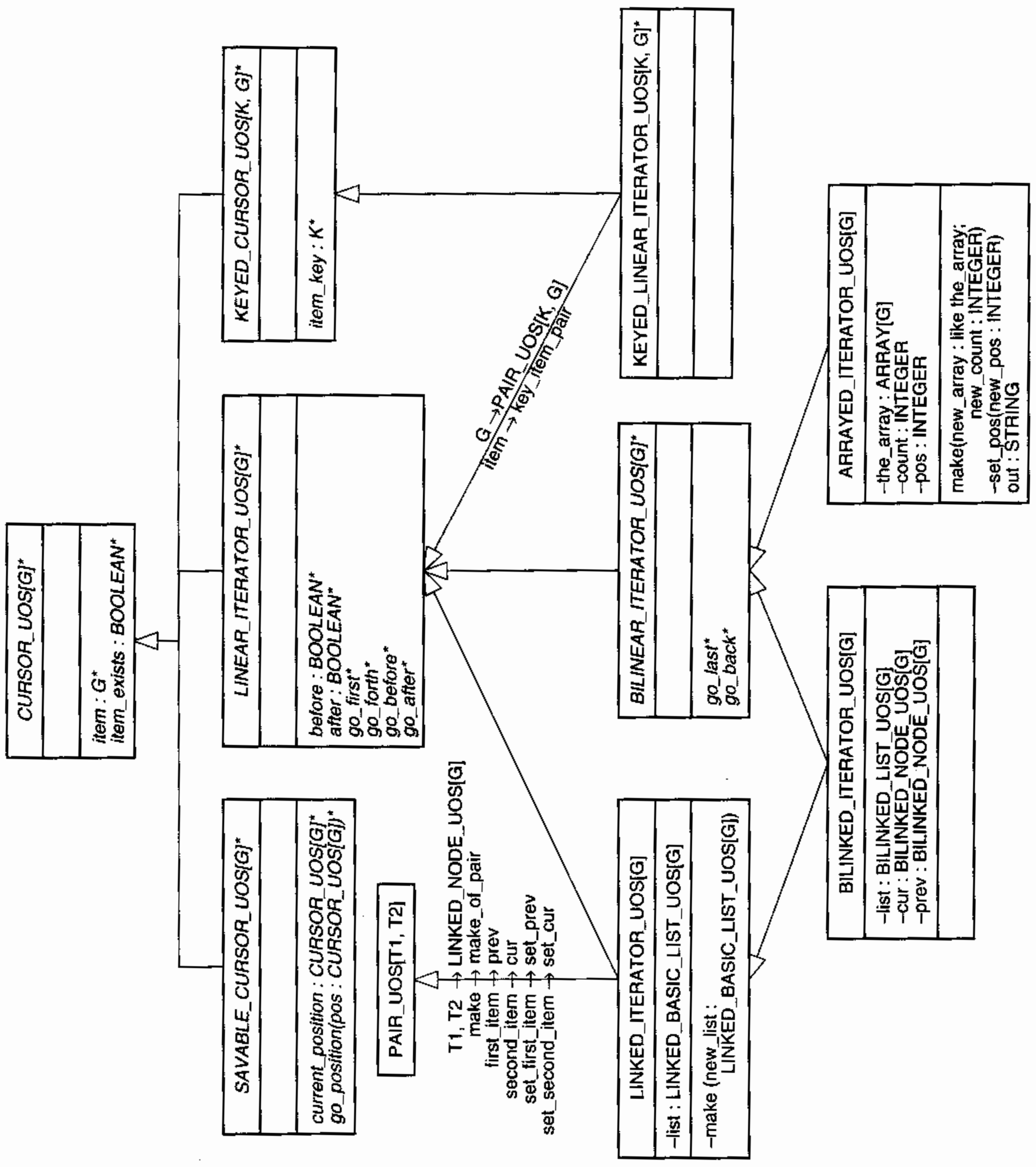

Figure 6.49. Iterators for linked lists

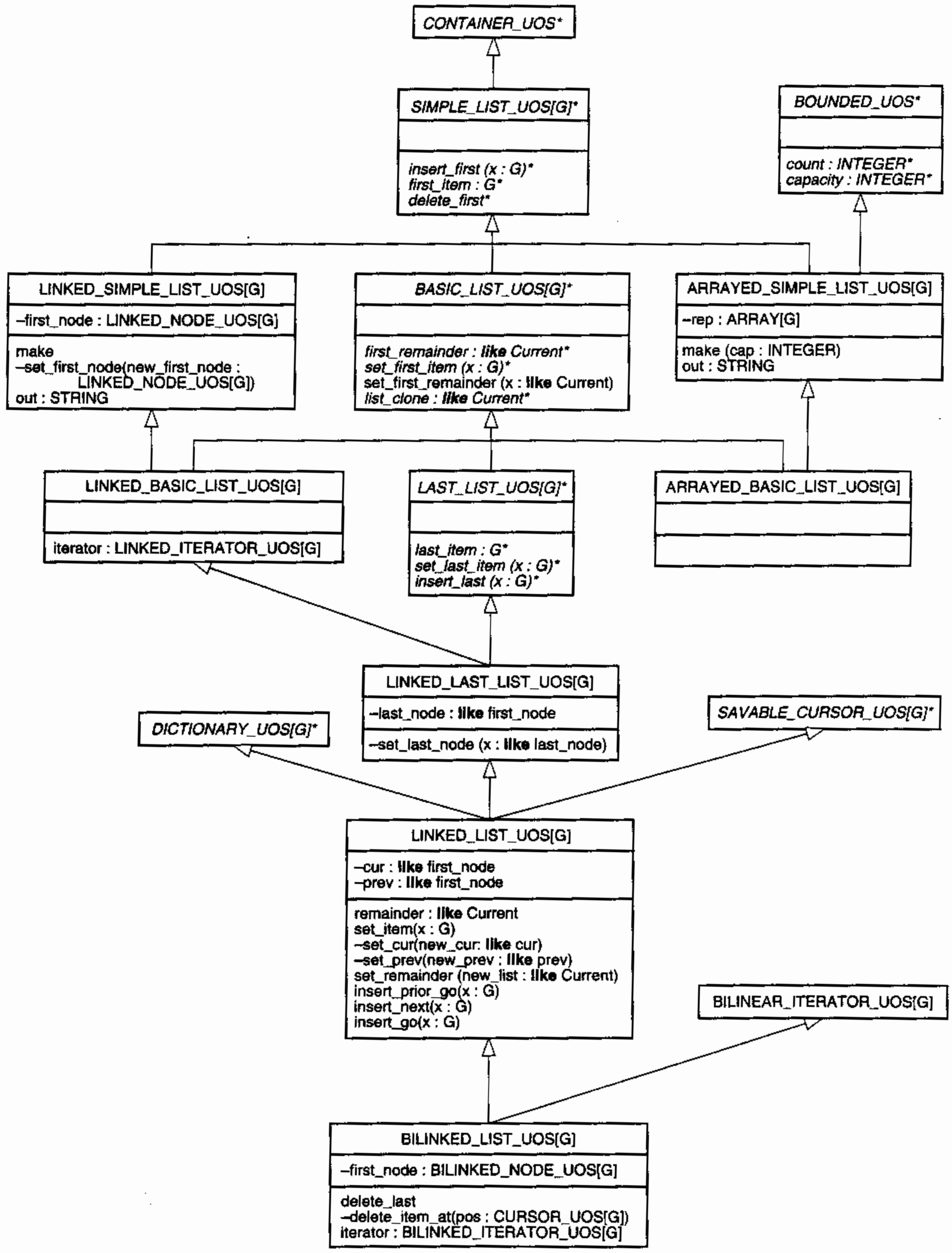

Figure 6.50. General linked list